IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Patent Application

Applicant(s): M.R. Betker et al.
Docket No.: 7-1-3-12-5
Serial No.: 10/072,529
Filing Date: February 8, 2002
Group: 2191
Examiner: Satish Rampuria

Title: Multiprocessor System with Cache-Based Software Breakpoints

---

APPEAL BRIEF

Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Sir:

Applicants (hereinafter referred to as "Appellants") hereby appeal the final rejection of claims 1-12 of the above-referenced application.

REAL PARTY IN INTEREST

The present application is assigned to Agere Systems Inc. The assignee, Agere Systems Inc., is the real party in interest.

RELATED APPEALS AND INTERFERENCES

There are no known related appeals and interferences.

STATUS OF CLAIMS

Claims 1-12 are pending in the application. Claims 1, 11 and 12 are the independent claims. All the claims stand rejected under 35 U.S.C. §103(a). The §103(a) rejection is traversed.

STATUS OF AMENDMENTS

There have been no amendments filed subsequent to the final rejection.

SUMMARY OF CLAIMED SUBJECT MATTER

The invention provides improved techniques for implementing software breakpoints in a shared-memory multiprocessor system (Specification, p. 3, lines 7-8).

Independent claim 1 is directed to a method for implementing software breakpoints in a multiprocessor system having a number of processors each coupled to a main memory. Each of the processors has an instruction cache associated therewith. An instruction for which a breakpoint is to be inserted is retrieved from a corresponding instruction address in the main memory, and a breakpoint code, e.g., a debug opcode, is inserted at the instruction address in main memory. After the breakpoint code is executed by a given one of the processors, the retrieved instruction is stored in the corresponding instruction cache for that processor, and a use-once indicator, associated with the instruction as stored in the corresponding instruction cache for that processor, is set. The use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

Advantageously, embodiments of the software breakpoint techniques of the present invention ensure that a specified debug opcode or other breakpoint code can remain present in the main memory of the multiprocessor system at all times, such that all of the processors of the system will reliably fetch and execute that breakpoint code even if one or more of these processors are resuming execution from the breakpoint (Specification, p. 3, line 27 to p. 4, line 3).

FIG. 1, for example, shows an example of an illustrative multiprocessor system in which the claim 1 can be implemented. In accordance with an aspect of the claim, each processor 102-i in the figure is associated with an instruction cache (Icache) 110-i. FIG. 2, moreover, illustrates the operation of one of the instruction caches, while FIG. 4 shows a state and flow diagram illustrating the operation of a multiprocessor system in accordance with the invention. The claimed retrieving of an instruction from the main memory, for example is embodied in step 426 in FIG. 4. The claimed retrieving of an instruction from the instruction cache and the handling of use-once and validity indicators is embodied in steps 416, 418 and 420. Also see the Specification at p. 9, line 3 to p. 10, line 2.

Independent claims 11 and 12 are directed to system and computer program product claims, respectively, corresponding to method claim 1.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

Claims 1-12 are rejected under 35 U.S.C. §103(a) as being unpatentable over U.S. Patent No. 6,848,097 (hereinafter "Alverson") in view of U.S. Patent No. 6,615,368 (hereinafter "Dunlap").

ARGUMENT

Appellants incorporate by reference herein the disclosures of all previous responses filed in the present application.

Rejection under 35 U.S.C. §103(a) as being unpatentable over Alverson in view of Dunlap Claims 1, 3, 4 and 9-12

For a valid §103(a) rejection, the combined references must teach or suggest all the claim limitations. Manual of Patent Examining Procedure (MPEP), Eighth Edition, August 2001, §2143. Appellants respectfully submit that this requirement has not been met with respect to claim 1.

Claim 1 sets forth:

> A method of implementing a software breakpoint in a multiprocessor system having a plurality of processors each coupled to a main memory, each of the processors having an instruction cache associated therewith, the method comprising the steps of:
>
> retrieving an instruction, for which the breakpoint is to be inserted, from a corresponding instruction address in the main memory;
>
> inserting a breakpoint code at the instruction address in main memory; and
>
> after the breakpoint code is executed by a given one of the processors, storing the retrieved instruction in the corresponding instruction cache for that processor and setting a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor, wherein the use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

In formulating the §103(a) rejection of this claim, the Examiner argues that several elements contained in the portion of the claim beginning with the words "after the breakpoint code is executed" are described by Alverson at col. 15, lines 45-54 (Final Office Action, p. 3). Appellants respectively disagree. The paragraph of Alverson cited by the Examiner states in its entirety:

> The routine begins at step 505 where a request is received from a target thread or from the debugger. The routine continues at step 510 to determine whether the request is a request from the debugger to create a breakpoint at a specified instruction within the executable code of the target. If so, the routine continues to step 515 to allocate memory for the instruction group to be generated, and then invokes the Generate OOL Instruction Emulation Group subroutine 515 for the instruction to be replaced. The routine continues at step 520 where the generated instruction group is installed in the allocated memory, and information about the created group is stored in an accessible location. The stored information will include a mapping from the original address of the replaced instruction to the address of the first installed instruction of the instruction group, and may include information about the breakpoint such as the condition for a conditional breakpoint or any other information to be supplied to the breakpoint handler when this breakpoint is encountered. The routine then continues at step 525 to replace the specified instruction in the target code with a BREAK instruction.

One skilled in the art will recognize that this portion of Alverson describes the replacement of a specified instruction within the executable code with a breakpoint instruction. Obviously, this type of replacement step must occur before the breakpoint can be executed. In claim 1, on the other hand, the steps following the words "after the breakpoint is executed" explicitly occur after a breakpoint instruction is encountered within the executable code. Consequently, Alverson and claim 1 describe entirely different steps. Alverson describes the configuration of breakpoint instructions, while the last clause of claim 1 describes steps that are performed after breakpoints are actually executed. Therefore Alverson fails to teach or suggest these elements of claim 1. Moreover, Dunlap does not correct this fundamental deficiency in Alverson.

What is more, the Examiner further argues that Dunlap's change of flow (COF) flag describes the use-once indicator and the validity indicator in claim 1 (Office Action, p. 4, citing Dunlap at col. 6, lines 42-44 and 46-48). Appellants again respectfully disagree. Dunlap at col. 6, lines 42-48 states:

> COF address register 330 detects the active COF flag and loads the next instruction address (in the new branch) for subsequent serial output to external devices (process step 375).
>
> In addition, the next instruction address is stored in IP 340 and is sent to instruction decode/dispatch logic 200, which uses it to fetch the next instruction from the new branch.

Dunlap's COF flag is "an internal flag . . . that is activated whenever any change in linear flow occurs, such as a conditional branch instruction, or a call, or an interrupt, in any of the functional units in the microprocessor" (Dunlap, col. 5, line 65 to col. 6, line 1). The COF flag "is sent to COF register 330 and to an external pin on microprocessor 100" (Dunlap, col. 6, lines 6-8). In contrast, the claimed use-once indicator is part of an instruction that is stored in a particular processor's associated instruction cache. In addition, Dunlap's COF flag is not "operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache." Dunlap, in fact, makes no reference to two different indicators analogous to the claimed use-once and validity indicators. Accordingly, the proposed reference combination fails to teach or suggest all the claim limitations of claim 1.

Furthermore, in stating the motivation for combining aspects of Alverson with Dunlap, the Examiner states on p. 7 of the final Office Action:

> Therefore, it would have been obvious to a person of ordinary skill in the art at the time the invention was made to incorporate the method of setting a flag for the software breakpoint in the memory as taught by Dunlap into the method of implementing software breakpoint [*sic*] in a shared memory system as taught by Alverson. The modification would be obvious because of one of ordinary skill in the art would be motivated to have flags in a software breakpoint method to provide an optimize [*sic*] technique for debugging as suggested by Dunlap (col. 1, lines 51-61).

Nevertheless, a *prima facie* case of obviousness can only be established if there is "some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings." MPEP §2143. Any such showing of obviousness "must be based on objective evidence of record" rather than "subjective belief and unknown authority." In re Sang-Su Lee, 277 F.3d 1338, 1343-44, 61 USPQ2d 1430 (Fed. Cir. 2002).

The Examiner's above-quoted argument lacks any basis in objective evidence of record that would motivate one skilled in the art to combine the references as suggested. Instead, the Examiner has apparently used improper hindsight by using the Appellants' teachings as a blueprint to hunt through the prior art for the claimed elements and combine them as claimed. The result is an argument to combine references that finds its motivation in advantageous aspects of the present invention, namely, cache-based software breakpoints. The Federal Circuit has repeatedly held that such an approach is "an illogical and inappropriate process by which to determine patentability." Sensonic, Inc. v. Aerosonic Corp., 81 F.3d 1566, 1570, 38 USPQ2d 1551, 1554 (Fed. Cir. 1996).

Dependent claims 3, 4, 9 and 10 are believed to be allowable for at least the reasons stated above with respect to claim 1. Independent claims 11 and 12 are system and computer program product claims, respectively, corresponding to method claim 1. Therefore, they are also believed to be allowable for reasons similar to those set forth above.

Claim 2

Dependent claim 2 is believed to be in condition for allowance for at least the same reasons as its base claim, independent claim 1. Moreover, claim 2 is believed to contain separately patentable subject matter over the proposed Alverson-Dunlap reference combination.

Claim 2 sets forth:

> The method of claim 1 wherein the instruction cache includes a plurality of sets of instruction information, each corresponding to a particular instruction, a given one of the sets of instruction information comprising the validity indicator, the use-once indicator, an instruction tag, and instruction data.

In formulating the §103(a) rejection of this claim, the Examiner argues on p. 7 of the final Office Action that all the limitations of the claim are taught or suggested by Alverson at col. 16, lines 24-40. Appellants respectfully disagree. Alverson at col. 16, lines 22-40 states:

> If it was decided in step 510 that the received request was not to create a breakpoint, the routine continues at step 540 to determine if the received request is a directive from the debugger indicating to begin execution of one or more target threads or to resume execution of one or more halted target threads. If so, the routine continues at step 545 to notify the target threads to begin or continue execution as directed. When execution is resumed after a halt due to a breakpoint, execution of the OOL Instruction Emulation Group for that thread will be performed by the breakpoint handler for the thread. After step 545, the routine continues to step 547 to wait for a notification from a target thread indicating that execution of the target thread has halted. In addition to execution halts resulting from breakpoints, target threads may halt for a variety of other reasons such as executing an invalid instruction, encountering a watchpoint, having executed a specified number of instructions (e.g., single-stepping), or by receiving a user-initiated manual break directive.

This portion of Alverson, unlike the claim, fails entirely to teach or suggest that instructions within an instruction cache comprise a validity indicator, a use-once indicator, an instruction tag, and instruction data. As a result, the proposed reference combination fails to teach or suggest all the limitations of the claim as required for a valid §103(a) rejection under MPEP §2143.

Claim 5

Dependent claim 5 is believed to be in condition for allowance for at least the same reasons as its base claim, independent claim 1. Moreover, claim 5 is believed to contain separately patentable subject matter over the proposed Alverson-Dunlap reference combination.

Claim 5 sets forth:

> The method of claim 1 wherein the use-once bit when set, being operative via the cache control logic to clear the validity indicator associated with the instruction as stored in the instruction cache after a single fetch of the instruction from the instruction cache, thereby automatically causes a miss between the instruction address and the instruction as stored in the instruction cache when the given processor attempts to retrieve the instruction from the instruction cache subsequent to the single fetch.

In formulating the §103(a) rejection of this claim, the Examiner argues on pp. 8 and 9 of the final Office Action that the limitations of this claim are taught or suggested by Dunlap at col. 6, lines 40-50. Appellants respectfully disagree. Dunlap at col. 6, lines 39-50 states:

> However, if during execution of the pending instruction, one of the components in functional units 250 determines that a change of flow (COF) has occurred, the COF flag is set (i.e., active) (process steps 360 and 370). COF address register 330 detects the active COF flag and loads the next instruction address (in the new branch) for subsequent serial output to external devices (process step 375).
>
> In addition, the next instruction address is stored in IP 340 and is sent to instruction decode/dispatch logic 200, which uses it to fetch the next instruction from the new branch. The next instruction then becomes the pending instruction and processing continues (process steps 365 and 355).

Appellants respectfully assert that this portion of Dunlap fails entirely teach or suggest the interaction of the use-once bit and validity indicator for the retrieving of instructions in the manner claimed. For example, Dunlap does not teach or suggest that a validity indicator is cleared subsequent to a single fetch of an instruction in an instruction cache so that that instruction is only fetched once. Instead, Dunlap describes the use of a COF flag to cause instructions to be fetched from a new branch in the executable code with no mention of indicators analogous to the claimed

use-once and validity indicators. The Alverson-Dunlap reference combination, therefore, does not teach or suggest all the limitations of claim 5.

Claim 6

Dependent claim 6 is believed to be in condition for allowance for at least the same reasons as its base claim, independent claim 1. Moreover, claim 6 is believed to contain separately patentable subject matter over the proposed Alverson-Dunlap reference combination.

Claim 6 sets forth:

> The method of claim 1 wherein the use-once indicator associated with the instruction comprises a single bit stored in a given set of the instruction cache.

In formulating the §103(a) rejection of this claim, the Examiner argues on p. 9 of the final Office Action that the limitations of this claim "are similar to those recited in claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above." Appellants respectfully suggest that such a rejection is untenable. The proposed reference combination fails entirely to teach or suggest all the particular limitations of this claim (i.e., that the use-once indicator comprises a single bit stored in a given set of the instruction cache) as required under MPEP §2143.

Claim 7

Dependent claim 7 is believed to be in condition for allowance for at least the same reasons as its base claim, independent claim 1. Moreover, claim 7 is believed to contain separately patentable subject matter over the proposed Alverson-Dunlap reference combination.

Claim 7 sets forth:

> The method of claim 1 wherein the validity indicator associated with the instruction comprises a single bit stored in a given set of the instruction cache.

In formulating the §103(a) rejection of this claim, the Examiner argues on p. 10 of the final Office Action that the limitation of this claim are taught or suggested by Alverson's FIGS. 4A and 4B. Appellants respectfully suggest, however, that neither the cited figures nor their associated text teach or suggest a validity indicator comprising a single bit like that claimed. As a result, the proposed reference combination fails entirely to teach or suggest all the limitations of claim 7.

Claim 8

Dependent claim 8 is believed to be in condition for allowance for at least the same reasons as its base claim, independent claim 1. Moreover, claim 8 is believed to contain separately patentable subject matter over the proposed Alverson-Dunlap reference combination.

Claim 8 sets forth:

> The method of claim 1 wherein the instruction for which the breakpoint is to be inserted comprises an instruction having a noncacheable attribute associated therewith.

In formulating the §103(a) rejection of this claim, the Examiner argues on p. 10 of the final Office Action that the limitations of this claim "are similar to those recited in claim 1 and rejected under the same rational set forth in connection with the rejection of claim 1 above." Appellants respectfully suggest that such a rejection is untenable. The proposed reference combination fails entirely to teach or suggest all the particular limitations of this claim (i.e., that the instruction for which the breakpoint is to be inserted comprises an instruction having a non-cacheable attribute associated therewith) as required under MPEP §2143.

In view of the above, Appellants respectfully submit that claims 1-12 are in condition for allowance.

Respectfully submitted,

Date: June 15, 2006

Michael L. Wise
Attorney for Appellant(s)
Reg. No. 55,734
Ryan, Mason & Lewis, LLP
90 Forest Avenue
Locust Valley, NY 11560
(516) 759-2722

CLAIMS APPENDIX

1. A method of implementing a software breakpoint in a multiprocessor system having a plurality of processors each coupled to a main memory, each of the processors having an instruction cache associated therewith, the method comprising the steps of:

retrieving an instruction, for which the breakpoint is to be inserted, from a corresponding instruction address in the main memory;

inserting a breakpoint code at the instruction address in main memory; and

after the breakpoint code is executed by a given one of the processors, storing the retrieved instruction in the corresponding instruction cache for that processor and setting a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor, wherein the use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

2. The method of claim 1 wherein the instruction cache includes a plurality of sets of instruction information, each corresponding to a particular instruction, a given one of the sets of instruction information comprising the validity indicator, the use-once indicator, an instruction tag, and instruction data.

3. The method of claim 1 wherein the instruction address comprises an instruction tag, an index and a block offset.

4. The method of claim 1 wherein the cache control logic is operative to compare portions of the instruction address to corresponding portions of instruction information as stored in the instruction cache and checks the validity indicator in determining if there is a hit or a miss between the instruction address and the instruction information as stored in the instruction cache.

5. The method of claim 1 wherein the use-once bit when set, being operative via the cache control logic to clear the validity indicator associated with the instruction as stored in the instruction cache after a single fetch of the instruction from the instruction cache, thereby automatically causes a miss between the instruction address and the instruction as stored in the instruction cache when the given processor attempts to retrieve the instruction from the instruction cache subsequent to the single fetch.

6. The method of claim 1 wherein the use-once indicator associated with the instruction comprises a single bit stored in a given set of the instruction cache.

7. The method of claim 1 wherein the validity indicator associated with the instruction comprises a single bit stored in a given set of the instruction cache.

8. The method of claim 1 wherein the instruction for which the breakpoint is to be inserted comprises an instruction having a noncacheable attribute associated therewith.

9. The method of claim 1 wherein at least a subset of the retrieving, inserting, storing and setting steps are implemented under the control of a debugger which interfaces with the multiprocessor system.

10. The method of claim 1 wherein the breakpoint code inserted at the instruction address in main memory comprises a debug opcode.

11. A multiprocessor system comprising:

a main memory; and

a plurality of processors each coupled to the main memory, each of the processors having an instruction cache associated therewith;

wherein an instruction, for which the breakpoint is to be inserted, is retrievable from a corresponding instruction address in the main memory;

wherein a breakpoint code is insertable at the instruction address in main memory; and

wherein subsequent to execution of the breakpoint code by the given processor, the retrieved instruction is stored in the corresponding instruction cache for that processor, and a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor is set, wherein the use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

12. An article of manufacture comprising a machine-readable storage medium for storing one or more software programs for implementing a software breakpoint in a multiprocessor system having a plurality of processors each coupled to a main memory, each of the processors having an instruction cache associated therewith, wherein the one or more software programs when executed implement the steps of:

retrieving an instruction, for which the breakpoint is to be inserted, from a corresponding instruction address in the main memory;

inserting a breakpoint code at the instruction address in main memory; and

after the breakpoint code is executed by a given one of the processors, storing the retrieved instruction in the corresponding instruction cache for that processor and setting a use-once indicator associated with the instruction as stored in the corresponding instruction cache for that processor, wherein the use-once indicator, when set for the instruction as stored in the instruction cache, is operative via cache control logic to clear a validity indicator associated with the instruction after a single fetch of the instruction from the instruction cache, such that subsequent attempts by the

given processor to access the instruction as stored in the instruction cache will cause the processor to retrieve the breakpoint code at the instruction address in main memory.

EVIDENCE APPENDIX

None

RELATED PROCEEDINGS APPENDIX

None